Technical Report
3761

A Fault Tolerant Integrated Circuit Memory

by

Anthony Francis Barton

Technical Report (Ph.D. Thesis)

April 1980

Computer Science
California Institute of Technology
Pasadena, California 91125

Submicron Systems Architecture Project

sponsored by

Defense Advanced Research Projects Agency
ARPA Order #3771

and monitored by

Office of Naval Research
Contract #N00014-79-C-0597

ACKNOWLEDGEMENTS

First I would like to thank my parents whose support and encouragement have been unfailing over the past 26 years. Their early teaching and later guidance have been the most important influences in my life.

At Caltech my chief mentor was Chuck Seitz, my advisor; I am grateful to him for suggesting the topic for this research and for his help with and interest in the project. Useful contributions were also made by Lennart Johnsson, Jim Kajiya, Carver Mead and Ivan Sutherland. I have also benefited from working, talking and relaxing with other members of the Caltech community too numerous to mention individually.

Acknowledgements for financial support go to the North Atlantic Treaty Organisation (studentship B/76/3003) and the Defence Advanced Research Projects Agency (contracts N00123-78-C-0806 and N00014-79-C-0597).

ABSTRACT

Most commercially produced integrated circuits are incapable of tolerating manufacturing defects. The area and function of the circuits is thus limited by the probability of faults occurring within the circuit. This thesis examines techniques for using redundancy in memory circuits to provide fault tolerance and to increase storage capacity.

A hierarchical memory architecture using multiple Hamming codes is introduced and analysed to determine its resistance to manufacturing defects. The results of the analysis indicate that substantial yield improvement is possible with relatively modest increases in circuit area. Also, the architecture makes it possible to build larger memory circuits than is economically feasible without redundancy.

CONTENTS

Chapter 1
INTRODUCTION

IC fabrication and yield

In a typical manufacturing process an array of half centimeter chips is fabricated on a 10 centimeter diameter wafer. The process is subject to inconsistencies resulting from impurities in the chemicals used, imperfections in the silicon crystal structure and the presence of dust. Circuits which fail to work correctly due to such inconsistencies are normally discarded.

The number of working chips divided by the total number built is called the yield. It depends upon the defect density of the process and the area of the circuit. For fault-sensitive circuits it is equivalent to the probability that a chip will have no defects.

The Poisson distribution can be used as an approximate model for relating defect density, area and yield. The basic distribution is:

$$f(x) = u^{x} e^{-u} / x!$$

where u is the mean and f(x) is the probability of there being x events. For integrated circuits the mean is Na where N is the defect density and a is the area of the circuit. The yield of a fault-sensitive circuit is then the probability of no defects:

$$f(0) = e^{-Na}$$

In this thesis the defect density will be expressed in terms of defects per million square lambda. Lambda is the term introduced

by Mead and Conway (1) for making design rules independent of feature size and, for nMOS, is equal to half the minimum line width in diffusion or polysilicon.

Economics of circuit size

The size of commercial integrated circuit chips is dictated by a tradeoff between the cost of the chip and the cost of using that chip in a system. The cost of the chip (Cc) is determined by adding the processing cost (Cp) of a silicon wafer to the cost of determining which chips on the wafer are working (Cw) then dividing the result by the average number of working chips per wafer (w):

$$Cc = (Cp + Cw) / w$$

The cost of using a chip in a system is the sum of the packaging cost and the cost of providing interconnection to other parts of the system. The latter includes part of the cost of printed circuit boards, power supplies, cabinets and wiring harness.

The cost of the chips increases with increasing circuit area. The number of chips on a wafer is inversely proportional to the area of the chip but, because the yield of a circuit decreases with increasing area, the number of good chips on a wafer decreases faster than the inverse of the area. Thus the cost of the chip increases by a larger proportion than the function.

The cost of building a system from unpackaged chips decreases with increasing chip area. Increasing the chip area, and function, results in a smaller number of chips necessary for the system. The building cost per chip is lower for larger systems

because the packaging and interconnection costs increase more slowly than the number of circuits due to economies of scale.

Chip area	a	2a	4a
Chips per wafer	200	100	50
Percentage yield	50	25	6.25
Chip cost	$0.25	$1	$8
System cost per chip	$1.25	$2	$9
Chips per system	1000	500	250
System cost	$1250	$1000	$2250

Table 1.1

The tradeoff between parts cost and assembly cost can be illustrated by the following simplified example. An electronics company builds a system using 1000 integrated circuits. Each processed wafer costs $25 and contains 200 circuits with a yield of 50%: a per circuit cost of $.25. The building cost is $1 per circuit for a total of $1.25 per circuit and $1250 for the whole system. In an attempt to cut costs, an analysis is performed to determine the system cost if the circuit function and area are doubled. This change reduces the number of circuits per wafer to 100 and the yield to 25% giving a new circuit cost of $1. The system cost is now $2 per circuit for 500 circuits or $1000. A further analysis is performed to determine the total cost if the circuit function is quadrupled. Now there are 50 circuits per wafer and a yield of 6.25% for a circuit cost of $8. The total system cost is $9 per circuit for 250 circuits or $2250. This example is summarised in table 1.1.

The above example assumes a high degree of regularity in the overall system. For less regular structures the cost changes could be different depending on the ease with which system parti-

tioning can be achieved at different integration levels.

In addition to the reduction in yield as circuits become larger, there is an increase in testing cost. It is necessary to test more circuits to find enough good ones and each test takes longer because the circuits are more complex. At current levels of integration the time required to test a circuit is approaching the level at which it will have a significant impact on circuit cost. It is therefore important to consider testing cost in any look at future economic trends.

Fault tolerant design

The equivalence between yield and the probability of no defects in today's designs occurs because correct operation of the circuit depends on the correct operation of every circuit element. This thesis is concerned with examining design strategies under which it is possible for some circuit elements to fail without causing failure of the overall circuit. Such designs will be referred to as **fault tolerant** and conventional ones as **fault sensitive.**

The term **redundant** will be used as a synonym for fault tolerant. A design which uses more components than necessary yet requires them all to work is considered to be non-minimal rather than redundant.

The ratio of the extra area of a redundant circuit to the area of the equivalent irredundant circuit is called the **overhead.**

During the 1950s a lot of work was done on trying to improve the

reliability of computers by adding redundancy to the designs. This work was inspired by the low reliability of the individual components. Von Neumann (2) showed that networks of high reliability could be built from components of low reliability. This result was not very useful in practice because the overhead was very large; for example von Neumann calculated that to build a system with 97.3% reliability from components with 99.5% reliability would require an overhead of 1000.

A similar result including protection against interconnection errors was achieved by Winograd and Cowan (3). Their work consisted of an extension of Shannon's work on information theory (4).

The advent of transistors resulted in several orders of magnitude improvement in component reliability and a corresponding decrease in interest in redundant design. Now the number of components on a chip has risen to the point where redundancy within the circuits would make possible a much higher level of integration. The difference from the previous situation is that the redundancy is necessary to protect against fabrication errors rather than just lifetime failure.

In the case of integrated circuits the reliability of the individual devices is very high but the large numbers of devices comprising circuits leads to a low yield. Similarly the occurrence of interconnection errors due to fabrication flaws is very low. Thus the overhead required to provide high reliability in integrated circuits can be expected to be much lower than that for the unreliable components of the 1950s.

The major difference between the reliability of discrete components and the devices on an integrated circuit is that the probability of failure of discrete devices is independent whereas neighboring devices on an integrated circuit are often affected by a single defect. This difference suggests that the overhead will be different for integrated circuits than for discrete components with equivalent failure rates and that different redundancy techniques should be used.

The parallel result to von Neumann's is that arbitrarily high yield can be achieved by using sufficient redundancy. However the theoretical results are not always applicable to the real world. An early result in this research was the demonstration that, with sufficient redundancy, one could build a one megabit memory chip with a yield within a very small fraction of 100%. The problem was that the circuit would be three meters square.

Expected benefits from fault tolerant circuit design

Fault tolerant design will increase yield at the expense of chip area. For today's industrial chip sizes, where the area cost of processed silicon is the dominant cost, this increase in yield brings a reduction in cost provided that the average number of working chips per wafer is increased. Thus the yield of the chip must increase by a larger proportion than the area.

For larger chip sizes, where the testing cost becomes important, increased yield can be valuable even if the number of good chips per wafer does not increase. Here the number of chips which are discarded after testing is reduced, thereby reducing the average

number of chips to be tested in order to find one which works. Provided that the redundancy is added in a manner which does not increase the time required to test one chip, the average testing time per working chip is reduced.

In addition to tolerating fabrication defects, some types of redundancy can also protect against failures which occur while a circuit is in use. Transient or "soft" failures, such as those due to alpha particles, can be tolerated as well as permanent ones. Protection against "soft" failures increases the reliability of the system. Reduction of "hard" failures increases hardware reliability and reduces maintenance costs.

The absence of a direct connection between function and yield of a redundant circuit means that more function can be put on a chip. Designers will then have greater flexibility in partitioning a system between chips. This flexibility will result in faster systems because it will be possible to reduce chip-to-chip communication, the most common performance bottleneck in electronic systems.

Scope of the project

Fault tolerance for integrated circuits is a very large subject because different error correction techniques are indicated for different applications. The choice of memory design for this project was because of the regularity of the structures involved. It was hoped that this regularity would permit mathematical analysis for providing some basic rules for fault tolerant design. The work does not assume the use of any special

processes such as PROM fuses or MNOS switches. This decision was made so that any designs examined could later be built to verify that the analysis was correct.

The published work on memory redundancy was found not to be closely applicable to the internal design of integrated circuit memories. It was based on existing types of core and integrated circuit memory and so was not useful in the search for new architectures for higher yield. The main differences were due to the fact that in design for high yield the redundancy has to protect against both fabrication defects and lifetime degradation whereas the previous work considered redundancy only for protection against lifetime degradation. In addition, the error correction circuitry is implemented within the same chip resulting in a tradeoff between the yield and correction capabilities of those circuits.

A simple model of yield

The yield of a fault tolerant circuit is hard to calculate because of the effect of partial damage. If a circuit is assembled from several components, each of which is partially working, the chance of the overall circuit working depends on the positions of the defects in the components. To simplify this problem, the early work on this project was performed using a model under which components were assumed to fail completely if they contained any defects. Redundancy was introduced by specifying that, out of a group of components, some given number have to work. The yield of a component is determined using the Poisson model and the area of the component. A example of input to the

program is given in table 1.1. All the cell definitions are in terms of cell size (in lambda) or sub-cells except for the last one, where redundancy is introduced by the clause "b64 6 of 7" meaning that one of the "b64" cells can fail without causing the "b256" cell to fail.

Program input	Meaning
b1 sides 36 31	single bit store
rdec sides 63 31	row decoder
cdec sides 36 120	column decoder
corner sides 63 120	corner element
b64 b1 64 rdec 8 cdec 8 corner 1	64 bit storage array
eccnode area 58800	Hamming encoder/decoder
b256 b64 6 of 7 eccnode 1	256 bit redundant store

Table 1.1

This model will be known as the naive, or pessimistic, model. The inherent pessimism in this model derives from the treatment of redundancy. The model is based on the assumption that if a sub-cell is not 100% working then it is totally unreliable. This assumption can easily be shown to be false by the example that a memory array can contain a single bad bit but be otherwise perfect. This model was chosen despite its pessimism because of the difficulties in exact calculation of yield.

References

(1) C. A. Mead and L. A. Conway, Introduction to VLSI Systems, Addison-Wesley, 1980.

(2) J. von Neumann, Lectures delivered at Caltech, 1952.

(3) S. Winograd and J. D. Cowan, Reliable computation in the presence of noise, MIT Press, 1963

(4) C. E. Shannon, Bell System Technical Journal, No. 27, pp 379-423, 623-658, 1948

Chapter 2

DISCUSSION OF POSSIBLE APPROACHES

Static and dynamic redundancy

Static redundancy is the technique of providing extra parts of a system which can be used as spares in the event of component failure. An example of static redundancy is the spare wheel in an automobile. When one of the tires in use fails, it can be replaced by the spare. The car will be inoperative while the replacement is being made but, provided no further punctures occur, will then continue to function normally.

An example of static redundancy in integrated circuit design is the IBM 64k RAM (1) which incorporates extra lines of bit stores which can be used to replace lines containing bad bits. During initial testing such lines are identified and, if there are enough spare lines, the addressing mechanism is set up to access only correctly functioning lines. If there are more faulty lines than extra lines the chip is unable to function as a 64k RAM. If there are fewer faulty lines than spare lines then the unused spare lines are reserved for later reconfiguration in the event of component failure during use.

Dynamic redundancy is the technique of building a system using more than the minimum possible number of components in such a way that some can fail without causing the complete system to fail even temporarily. An example would be the jet engines in a large aeroplane. The aeroplane is designed to survive the loss of power from at least one of the engines, even at critical times

such as takeoff.

There would appear to be no current examples of dynamic redundancy in commercial integrated circuit design, but it is often used at higher levels in computer design.

For integrated circuit design both approaches to redundancy have appealing features. Static redundancy is relatively cheap to implement because the number of extra components required is equal to the number of faults to be tolerated. It is highly effective against fabrication flaws because configuration can be done during initial testing, but it is less satisfactory for component failure because external error checking is necessary to detect such failure and the circuit must be removed from service for reconfiguration. A major problem with static redundancy is how to distinguish between transient errors and component failures and hence how to decide when reconfiguration is necessary.

Dynamic redundancy is relatively expensive to implement because it is based upon the encoding of information, and the number of extra components required is larger than the number of faults to be tolerated. A single-bit error corrector which provides k corrected output bits from n input bits requires that n-k be greater than or equal to the logarithm to the base 2 of n. For a full discussion of error correcting codes see Peterson and Weldon (2). Dynamic redundancy is equally effective against fabrication flaws, component failures and transient errors. Using the properties of error correcting codes it is also possible to design the circuits in such a way that they are partially self-

diagnostic.

Methods of implementing redundancy

In addition to the choice between static and dynamic redundancy, it is also necessary to choose the level at which the redundancy is to be implemented. Level here refers to the size of the unit which is to be replicated; the unit could be a transistor or wire at the lowest level or a complete circuit at the highest level. In general, static redundancy techniques are applicable at any level whereas different dynamic redundancy techniques are better for different levels.

Quadded logic (3) is a low-level dynamic redundancy scheme under which the basic gates are replicated four times and interconnected in such a manner that errors are corrected close to their point of origin. This technique was developed for circuits built from discrete components and depends for its effectiveness on the probability of failure of neighboring elements being statistically independent. Because integrated circuit failures often result in the failure of a group of devices this technique does not lend itself very well to integrated circuit technology.

Error correcting codes can be used to provide dynamic redundancy. In the case of integrated circuits it is necessary to take into consideration the chip area required for the encoder and decoder. The complexity of these circuits is such that the codes are not practicable for low-level redundancy.

A hybrid form of redundancy can be achieved by providing circuits which monitor irredundant circuits and select the ones which

appear to be the most reliable. This approach could be regarded as static redundancy with continuous reconfiguration or as dynamic redundancy. Since the external effect of such circuits is the same as for dynamic redundancy they will be considered to perform dynamic redundancy. As in the case of error correcting codes, this technique requires complex circuits for implementation and is not suitable for low-level redundancy.

What others have done or plan to do

While the industry standard dynamic RAM was still 16k, IBM introduced a new computer system, the 8100, which uses 64k dynamic RAMs (1). They achieved this capacity by using a large die size and incorporating static redundancy into the design. The chip features spare bit lines which can be used in place of lines which are wholly or partially defective. Such defective lines must be detected during initial testing and disconnected by means of fusible links in the metal layer. A replacement line is connected by similar means. This technique is not sufficient to correct major faults but does offer relatively cheap protection against fabrication flaws affecting small numbers of bits. In addition, a circuit in which a similar fault occurs during service can be reconfigured provided that sufficient spare parts remain unused.

In the middle sixties, researchers at Texas Instruments introduced a discretionary wiring system for connecting working circuits on a wafer (4). Circuits were built in the normal way, then the working ones were identified on each wafer. A number of masks, corresponding to extra layers of metalization, were then

designed to combine the working circuits into the required system. This technique can protect against fabrication errors only and so is an example of static redundancy.

Discretionary wiring has not been a success because of the expense of designing special masks for each wafer. A modified scheme was proposed at Hughes Aircraft Company under which an extra metal mask was introduced (5). This mask was used to map the actual positions of good circuits into a predetermined pattern. This scheme required only one specially designed mask for each wafer but was still not cost-effective. There was also too high a loss of wafers due to defects in the metal layers used to interconnect the working circuits.

Researchers at Honeywell introduced a technique called superchip which was designed to avoid the problems encountered with discretionary wiring (6). Cells containing memory arrays and a bus are built simultaneously on a wafer. The cells are tested individually and those that work are connected to the bus using PROM switches. Bus addresses are also assigned to the cells by means of PROM devices. This scheme does not seem to have been very successful, probably due to low yield of the bus. Superchip, like discretionary wiring, is an example of static redundancy, offering protection against fabrication flaws only.

A further development called adaptive wafer scale integration (AWSI) was proposed at Actron, a division of McDonnell Douglas Corporation (7). Where superchips use PROM fuses, AWSI uses MNOS non-volatile, electrically alterable switches. This approach allows reconfiguration during use. If such reconfiguration is

provided internally by the use of special monitoring circuits dynamic redundancy can be implemented. AWSI, like superchip, uses a large bus and is likely to suffer from low yield of that bus.

J. I. Raffel of MIT Lincoln Labs has been investigating a technique using MNOS switches but a different bus structure (8). He proposes the use of wafers containing circuits of MSI to LSI complexity with a number of horizontal and vertical bus wires between each row and column of circuits. Some of these wires would run the entire length of the wafer to provide long distance communication. Others would be divided into sections to provide more local signals. Every intersection of two bus wires would be provided with an MNOS switch to permit electrical interconnection for routing of signals.

Raffel's structure is a very general one. It could be used to implement static redundancy by putting irredundant circuits on the wafer and doing all testing and configuration from outside. By making some of the circuits redundant it would be possible to use a mixture of static and dynamic redundancy. Alternatively, a form of dynamic redundancy could be provided by incorporating circuits on the wafer to perform reconfiguration.

A major omission from Raffel's proposal is that, so far at least, he has failed to address the problem of defects in the configuration circuitry and the bus structure. The failure rate of the programmable links and the wires in the bus structure should be investigated. Then an analysis should be performed to determine whether the number of such failures would be within

acceptable limits.

Of the techniques described above, only the relatively modest one used by IBM has been a commercial success. A common trait in the others which might indicate the reason for their failure is the reliance on a complicated bus structure. In the case of the TI and Hughes systems, the buses are added after testing the individual circuits. Therefore it is possible to introduce new defects at this stage. The other techniques all rely on large bus structures, occupying large areas of silicon, to be defect free and, moreover, require all defective circuits to fail in such a manner that they do not affect the bus.

References

(1) E. F. Pierce, Memory, Think, pp 4-8, November 1978.

(2) W. W. Peterson and E. J. Weldon, Error Correcting Codes, MIT Press, 1972.

(3) J. G. Tryon, Quadded Logic, Redundancy Techniques for Computing Systems, pp 205-228, Spartan Books, 1962.

(4) J. W. Lathrop et al, Proceedings of the IEEE, Vol. 55, No. 11, pp 1988-1997, 1967.

(5) D. F. Calhoun, AFIPS, Vol 35, pp 99-109.

(6) J. C. Hunter, pp V450-469, Proceedings of the Symposium on Advanced Memory Concepts, Stanford Research Institute, June 1976.

(7) W. A. Geideman and A. L. Solomon, Wafer Integrated Semiconductor Mass Memory, presented at the International Telemetering Conference, 1978.

(8) J. I. Raffel, On the Use of Nonvolatile Programmable Links for Restructurable VLSI, Proceedings of the Caltech Conference on VLSI, pp 95-104, 1979.

Chapter 3
THE HIERARCHICAL REDUNDANT MEMORY (HRM)

Derivation of the HRM

The first approach taken was to consider a design where the circuit was composed of two parts: one which implemented the redundancy and one which performed the primary function of the circuit. It seemed that, since the redundancy would correct errors due to faults in the main section, the yield of the total circuit would be more closely related to the size of the error correcting portion than to the size of the whole circuit.

It was found that adding error correction circuitry to a standard memory array was not cost effective. The length of the data and select wires meant that defects had a high probability of affecting large numbers of bits. Multiple error correction schemes were necessary to protect against these failures. The number of extra bits necessary for storing the encoded bits and the low yield of the multiple error correction circuits resulted in high overheads with little or no yield improvement.

In an effort to reduce the effect of defects the storage arrays were divided into sections to reduce the number of bits affected by row and column failures. The resulting memories had good yields only when the arrays were small enough that single bit error correction could be used. This limit to the size of the arrays set a limit on the capacity of the whole circuit; the limit was removed by combining several such circuits into a larger circuit and using another stage of error correction. That

architecture was named the hierarchical redundant memory.

Description

The HRM datapath is in the form of a tree. Each node and its descendants form a sub-memory (figure 3.1). Each arc of the tree represents a one-bit wide bi-directional data path. Each leaf node consists of an irredundant memory array with address decoders. The remaining nodes each contain an encoder/decoder for an error correcting code (ECC) and an address multiplexor. Figure 3.2 shows a non-leaf node for the Hamming (7,4) code. The HRM as described stores one bit at each address; by omitting the multiplexor from the root node the HRM can be used to store the same number of bits organised as k bit words.

This thesis assumes the use of quasi-perfect codes, such as Hamming codes, because they lend themselves to relatively easy analysis. Other, less efficient, codes might turn out to be more suitable for the HRM if they have very simple encoder/decoders.

The branching ratio of the tree depends on the particular ECC used in the nodes. All the nodes at one level would normally use the same ECC and, in the following, it will be assumed that the same one is used at all levels of the tree. Where the code is an (n,k) code, ie. it produces k corrected bits from n input bits when decoding, the tree has a branching ratio of n. The number of bit stores in a sub-memory is thus n times the number at the next lower level, while the number of addressable bits is only k times the number of addressable bits at the next lower level.

In a read access of the memory one bit is read out from each leaf

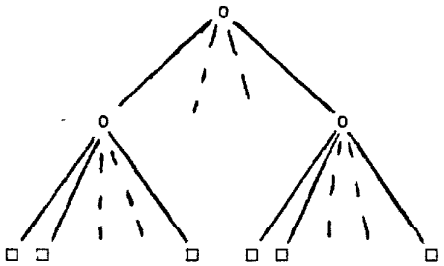


Figure 3.1

node and passed up to the parent node. Each non-leaf node receives n such bits passed up from its children, passes them through its ECC decoder, and obtains k "corrected" bits. Whether these bits are in fact correct depends upon how many of the incoming ones were correct. One of the corrected bits is selected by the multiplexor, according to the given address, and passed up to the next level.

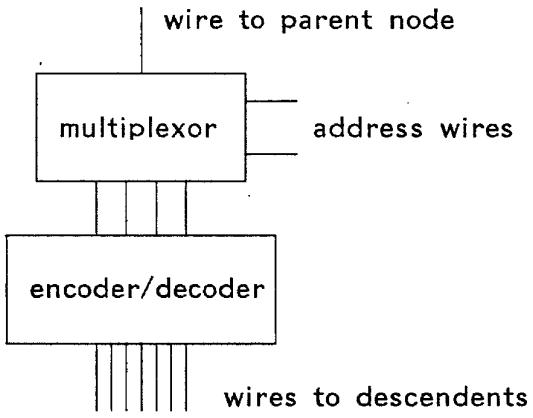


Figure 3.2

A write access can be performed only after a read access on the

same address. Thus every write access is really a read-modify-write cycle and the HRM is either implemented as such internally or always used this way. After the read has been performed, the new datum is presented at the root node. The multiplexor then overwrites one of the k corrected bits with this new value. The k bits are passed through the encoder, giving n encoded bits. These bits are passed to the n children and the process is repeated down through the tree.

Rationale

The HRM was chosen as an example of a fault tolerant memory because it achieves multiple error correction with simple decoders, has small critical areas and conforms with memory architectures suggested by an examination of the physics of computation.

The use of the tree structure means that, even if the individual error correctors can correct only single bit errors, certain types of multiple bit errors on a single access can be tolerated. At any node in the tree it is necessary for at least two errors to occur in order for an error to be passed up to the next level. For example in a memory with two levels of redundancy using the Hamming (12,8) code, each access involves 144 bits. The worst case under which a given bit address can operate correctly is when at the higher level decoder one bit is unreliable due to all 12 bits which contribute to it being unreliable and in the other 11 cases one bit is bad in each group. Thus in this hypothetical example the use of multiple single bit error correctors would give the correct answer despite the presence of 23 bad bits. At

the opposite extreme is the case where two errors in each of two groups of bits causes an uncorrectable error. These examples assume that the error correctors are functioning perfectly but, since error corrector failure is equivalent to, and indistinguishable from, failure of all the bits in the associated leaves, this assumption is not unreasonable.

Because the error correctors are simple they are relatively small. Furthermore the only part of the memory that is absolutely crucial is the root node. This node represents a very small proportion of the whole circuit and can be built with conservative design rules, or replicated and coded, without significantly increasing the size of the overall chip. Similarly the lower level decoders must be working if the sub-trees of which they are roots are to be useable. A good design approach would therefore be one where the leaf memory arrays are built to the minimum tolerances recommended for the fabrication process and the higher nodes in the tree are built with successively more conservative rules or increased replication.

It has been demonstrated by Mead and Rem (1) that, for many relevant cost functions, the optimum memory designs use a hierarchical structure with a relatively low branching ratio. Apart from the absence of redundancy this structure is essentially the same as that of the HRM. The basic tree structure is thus indicated both for redundancy and for a low speed-power product.

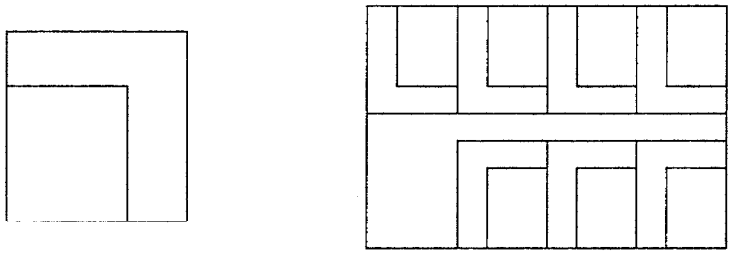
Layout for the HRM

Although no HRM has been built, considerable thought has been given to how one would be laid out on silicon; partly with the intention of estimating the size of the necessary circuits and partly to provide actual circuits for the experiment described in chapter 4. The circuit designs presented here use nMOS depletion load technology and the design rules described in Mead and Conway (2). Where circuit layouts are given in figures the colors are according to the Caltech convention of green for diffusion, red for polysilicon, blue for metal, black for contact cuts and yellow (or sometimes black to improve visibility) for implant.

The first design produced was that for the leaf arrays. A layout was made for a rectangular array of static memory cells with the number of cells in each direction being a power of two. Two sides of the array had conventional row and column address decoders, with the column decoder also providing the data input and output.

The other circuit layouts produced were for the various parts of the non-leaf nodes. The design for a Hamming code decoder involved only exclusive NOR gates and AND gates in a rectangular grid and could easily be parameterised for the family of codes although the one actually designed was for the (7,4) code. The encoder was made by just taking the appropriate part of the decoder. It would have been possible to combine the two parts into a single circuit by sharing the common section but the extra circuitry to accomplish the sharing would have offset most, if not all, of the saving. Moreover, by keeping the two parts

separate it was possible to improve the fault tolerance slightly because certain types of errors in the encoder resulting in single bit errors are still corrected by the decoder. The third part of the non-leaf node, the multiplexor, was adapted from a memory cell and the column address decoder.



Leaf HRM with one level of redundancy

Figure 3.3

An overall layout was considered for a complete HRM design using the (7,4) code. This code is attractive because it means that seven leaf arrays and one non-leaf node can be grouped in a rectangular array provided that the non-leaf node is no larger than each leaf node. Such an arrangement is shown in figure 3.3. It turns out that, for this code, the non-leaf node can be made significantly smaller than a 64 bit leaf array. At the next level up, seven of these groups and a non-leaf node are to be laid out. This time, if the rectangular grid is repeated, there is eight times as much area available as at the previous level. Ample space is now available for using conservative design rules, as described above, or for redundancy through replication and voting.

Testing the HRM

At first glance it would appear that the HRM, being more complex than traditional memories, would require more time to test. If so it might no longer be true that increasing the yield would decrease the cost of testing. In fact the HRM can be tested more quickly than conventional memories due to the properties of its architecture.

One property of the HRM which reduces testing time is that it can never have only one bad bit. The method by which the redundancy is implemented is such that, if any bit is bad, at least c/m bits will be bad where c is the number of bits in the memory and m is the number of bits in each leaf node. The presence of bad bits can therefore be detected in time proportional to the size of the leaf arrays rather than the size of the whole memory.

Testing time for misaddressing and pattern sensitivity is less for the HRM than for conventional designs because of the hierarchical structure. The testing strategy is to start by testing the root node by changing only the address bits used in the root multiplexor. Once the root has been fully tested the next level down can be tested in a similar way until the whole tree has been tested. This strategy, made possible by the architecture, is much better than is possible with conventional designs where this sort of partitioning is not possible.

Yield crossover point

The most important result from the use of the naive model was the notion of the yield crossover point (YCP). Where a non-leaf node

uses an (n,k) error correcting code (ECC), an upper bound on its yield (Y) can be given, as follows, in terms of the yield of its n children (y) and the maximum number (t) of errors which can be corrected by the code:

$$Y = \sum_{i=0}^{t} {}^{n}C_{i} \; y^{n-i} \; (1-y)^{i}$$

where ${}^{a}C_{b}$ is the number of combinations of a objects taken b at a time

The YCP is that value of y for which Y = y, ie the yield of a node such that its parent shall have the same yield. Y is an upper bound on the composite yield, because the above calculation should really include a factor representing the yield of the contents of the parent node. Thus the above definition of the YCP results in a value lower than is required for practical applications. Some values of the YCP for Hamming codes are given in table 3.1.

Code	YCP
(3,1)	50%
(5,2)	86.9%
(7,4)	94.2%
(12,8)	98.3%
(21,16)	99.5%

Table 3.1

For values of y less than the YCP, Y falls off very quickly; it is therefore impractical to build leaf nodes with yield much

lower than the YCP. Given the number of faults per unit area for a process and an ECC, it is thus possible to calculate the maximum area, and hence the maximum number of bits, for useable leaf nodes for that process and ECC. The size of the leaf node indicates the number of levels required in the tree to provide a memory of the desired size. The approximate area of the overall circuit can then be calculated.

References

(1) C. A. Mead and M. Rem, Cost and Performance of VLSI Computing Structures, Proceedings of the 3rd USA-Japan Computer Conference, pp 462-467, 1978.

(2) C. A. Mead and L. A. Conway, Introduction to VLSI Systems, Addison-Wesley, 1980.

Chapter 4
STATISTICAL YIELD MODELING

Limitations of the naive model

The naive yield model could not be used for a detailed study of the HRM unless it could be shown to be sufficiently accurate. An experiment was performed to calculate the probability of a circuit working while having a pattern of faults which the naive model treated as fatal. For this purpose a one level HRM using the Hamming (7,4) code was examined. Each of the leaf arrays contained 64 bits, resulting in a 256 bit memory. Under the naive model the root and at least six of the leaves had to be fully functional. The only other way in which the memory could work was if up to 64 individual bit stores failed such that they were not all in the same array and that no two were in corresponding positions of different arrays. The calculated expression for this probability was

$$P = \sum_{i=2}^{64} a^{7} b^{448-i} (1-b)^{i} C_{i}^{64} (7^{i} - 7)$$

where a is the yield of the address decoder in a leaf node and b the yield of the individual bit stores. For a derivation of this expression see appendix B. The naive model indicated that 0.56% of circuits would fail. The value of P obtained after taking the values of a and b from the naive model was 0.28%. So, for this example, half of the failures found by the naive model were directly attributable to the pessimism of the model.

It is interesting to note that this limitation does not apply if the leaf arrays are replaced by shift registers. A shift

register has the property that, if one bit store is bad, it corrupts all data as they pass through. A HRM built from shift registers could thus be evaluated using the naive model.

Purpose of statistical modeling

The excessive pessimism of the naive yield model is due to the fact that it is based on the incorrect assumption that a sub-tree containing a fault is completely unreliable. It had been hoped that it would prove sufficiently accurate because, as described above, it was the best approximation to the real situation that lent itself to an exact calculation. Since it did not prove accurate enough, it became necessary to use a model which was an accurate representation of how the HRM was affected by faults but was applied only to a number of random fault patterns.

The statistical experiment was composed of two parts. The first part was used to gather statistics on the relative frequency of faults affecting various numbers and groupings of bits of memory in leaf nodes of the HRM. In the second part HRMs were "built" using leaves with faults assigned according to these statistics.

The statistics gathering program

The program was written so as to draw a given number of random circles with an appropriate distribution on a circuit design. The circles were then interpreted as manufacturing defects. Visual inspection of the drawings was used to determine how much of the circuit was affected by each "defect". In addition this experiment gave a measure of what proportion of defects materi-

ally affect the performance of the circuit.

The model for distribution and interpretation of the circles was made to be as simple as possible without departing too far from reality. It was important to concentrate on those factors which are likely to persist in the future rather than those exhibited today. The available data seemed to indicate that the biggest problem was dust, both on the wafer during manufacturing and on masks while making copies. By assuming the use of direct electron beam writing the problem could then be reduced to one of failure to expose the photoresist wherever there was dust on the wafer. Thus from the nature of the process it is possible to deduce whether a given layer is susceptible to opens or shorts. The circles were drawn with a uniform distribution of position and layer and a distribution of size such that the average number of circles of a given area was inversely proportional to that area.

The process used for the experiment was nMOS with depletion loads. This process uses five masks. It was assumed that the probability of defects in the implant layer causing faults was low enough to be ignored and that double masking would be used for contact cuts, again making the probability of faults insignificant when compared with the other layers. Of the remaining layers metal and polysilicon were susceptible to opens and diffusion to shorts.

Figure 4.1 shows an example of a circuit consisting of four static RAM cells with two errors on it. The error in diffusion (green) causes two cells to be linked together in such a manner

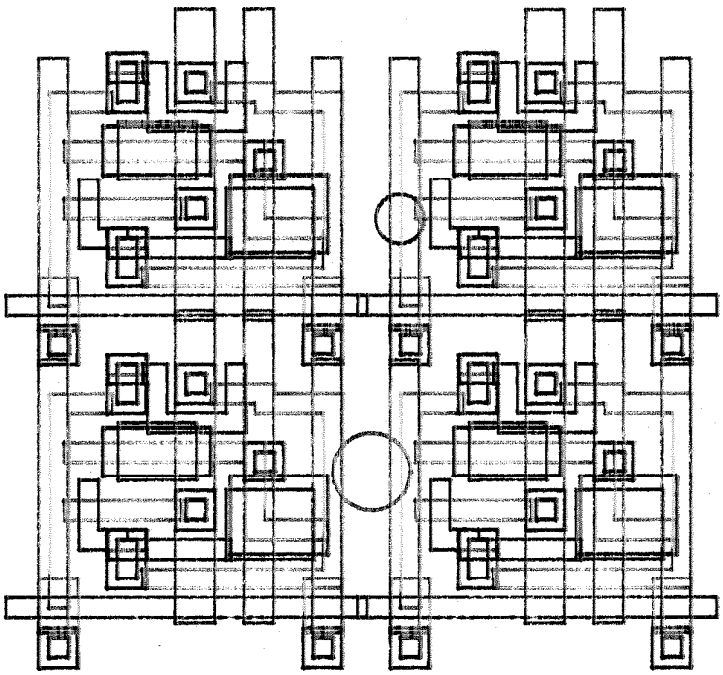


Figure 4.1

that each cell always stores the complement of the other. Depending upon the addressing scheme used this error causes the loss of either one or two bits. The error in metal (blue) causes a cut in a data line. This defect will result in the loss of all bits in the column which are on the far side of the fault from the data line driver.

Results from the circles program

The circles program was run several times using a 64 bit leaf array. In all 800 circles were drawn and examined and, of these, 556 had no effect, 9 caused power-to-ground shorts and the

remaining 225 caused varying numbers of bits to be inaccessible or unreliable. These were divided into a number of modes of failure and, in particular, a distinction was made between failures due to faults within the storage array and those in the address decoders.

Failure mode	% for array	% for decoder
Single cell	43%	-
Pair of cells	7%	-
Part row	11%	-
Part column	28%	-
Single column	3%	37%
Single row	-	7%
Multiple columns	-	23%
Multiple rows	-	12%
Whole array	8%	21%

Table 4.1

There were six failure modes due to defects occurring within the memory cells. These were single bit failure, double bit, part row, part column, whole column and power-to-ground short. Single bit failure occurs when a defect causes damage only to logic within the cell and is usually a defect in the diffusion or polysilicon layer. Double bit failure results from a diffusion bridge between two neighboring cells, causing them always to store complementary values. Part row failures occur when the select line is cut inside a cell. Then the cell itself and all cells beyond it are affected. Such a failure, if it occurs in the cell nearest to the row decoder, can affect a whole row but this case cannot be identified without knowing which cell is involved and so is not considered a separate mode. Part column

failure is the equivalent failure in a vertical wire; data, data bar, ground or power. In the case of the first three the cells above the one in which the error occurred are affected but the power line is driven from above and so, if it is cut, the cells below are affected. Whole column failure is the case when both the power line and one of the other three are cut.

It is hard to assess the effect of power-to-ground shorts and they are arbitrarily assumed to affect the whole leaf array but not the rest of the circuit. This assumption is on the basis that if they are not bad enough to burn themselves out they will not consume enough power to affect more than the local circuitry and that if they do burn out only a relatively small area will be affected by the heat and debris.

For the decoders there were five failure modes: single column, single row, multiple row, multiple column and whole array. The reasons for the different modes were less distinctive than for the memory cells but, in general, they resulted from cut wires. The most serious errors resulted from cuts in the power, ground or global data wires and the moderately serious ones from cuts in address lines.

"Tree" program

With this program a user can "design" a HRM by supplying the dimensions of the leaf arrays, the error correction code to be used in the non-leaf nodes and the number of levels in the tree. The user requests the program to "build" some number of them and the program does so using leaves with errors randomly assigned

according to the results collected from the first experiment. The program indicates how many of the memories had errors and how many bits were defective in each case. Unlike the circle drawing program which was not at all specialised and could be used for investigating the failure modes of any circuit, this program is specific to the HRM and would not be applicable to any other architectures.

The program consists mainly of a leaf generation routine and a combination routine. The leaf generation routine creates an array of bits each of which is set to 0 if the corresponding bit is deemed to be working and 1 if it is deemed to be useless. The decision on which bits work is reached by means of a random number generator and a table of failure modes and their frequencies derived from the first part of the experiment. The combination routine allocates a new array of bits to represent a sub-tree of the memory and fills in the bits according to the leaves or lower level sub-trees of which it is composed. For an (n,k) code there will be k times more bits at each level than at the one beneath and each bit will be set to 1 if more than one of the n bits from which it is derived are 1. In addition the array is set to all ones if the non-leaf node is deemed to have any defects. This treatment is slightly pessimistic because there are ways in which a non-leaf node can have defects and still be useful but they are rare enough to be ignored.

Chapter 5
RESULTS OF THE STATISTICAL EXPERIMENT

Caveat

In considering the following results it is important to remember that no number should be taken in isolation. There is no intention to prove that a memory of a given size could be built with a given yield. The semiconductor manufacturers are reluctant to talk about the yields which they achieve for their production circuits; therefore it is hard to estimate the typical industrial defect densities which must be known in order to discuss what yields could be achieved for the HRM. Furthermore no direct comparison is possible with industrial yields because the design rules and feature size are different. For these reasons all comparisons are with irredundant memories built from the same basic static RAM cell as that used for the HRM.

Yield as a function of defect density

The Poisson model of yield indicates that yield of irredundant circuits declines exponentially with increasing defect density. The more complex models, which correct some of the inaccuracies of the Poisson model, predict higher yields than the Poisson model but with a similar shaped curve. For the HRM the shape of the curve is influenced greatly by the way in which the redundancy is implemented. The curve is related to the result of convolving an exponential with itself but differs in that the original curve is more complex than an exponential and the combinational operation is not a pure convolution. A pure convolution

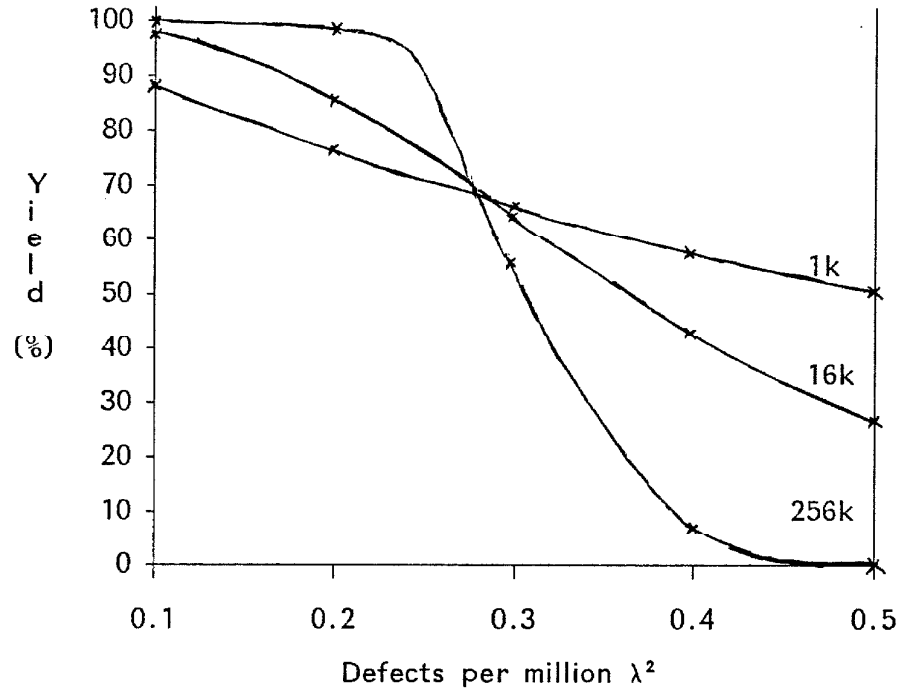


Figure 5.1

would result if all failures affected only individual bit stores.

Figure 5.1 shows a graph of yield against defect density for three related circuits. The first circuit is a 1k RAM with no redundancy. The other two circuits are HRMs using the 1k RAM as a leaf node. Both HRMs use the Hamming (7,4) code; one with 2 levels of redundancy resulting in a 16k memory, the other with 4 levels for 256k. The graph shows that, as predicted in chapter 3, the yield increases with increasing number of levels for low defect density then decreases with increasing number of levels for higher defect densities. The crossover point is at a lower yield than the YCP because the derivation of the YCP included an

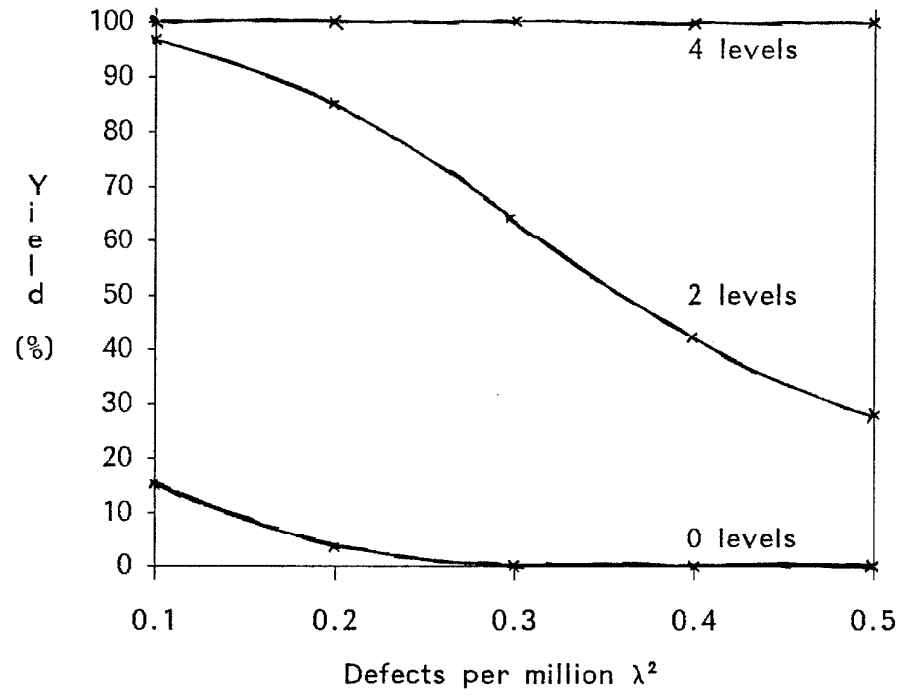


Figure 5.2

assumption that a defect in a leaf node resulted in total failure of that node.

The intention in figure 5.1 is to show the way in which the composite yield of an HRM is affected by the yield of the parts. A more spectacular demonstration of the yields which can be achieved using the HRM is given in figure 5.2 by plotting the yield of three memories of the same size. This graph shows a direct comparison of the yields possible for a 16k RAM at different defect densities using no redundancy, two levels and four levels. The two level memory is the same as that used in figure 5.1.

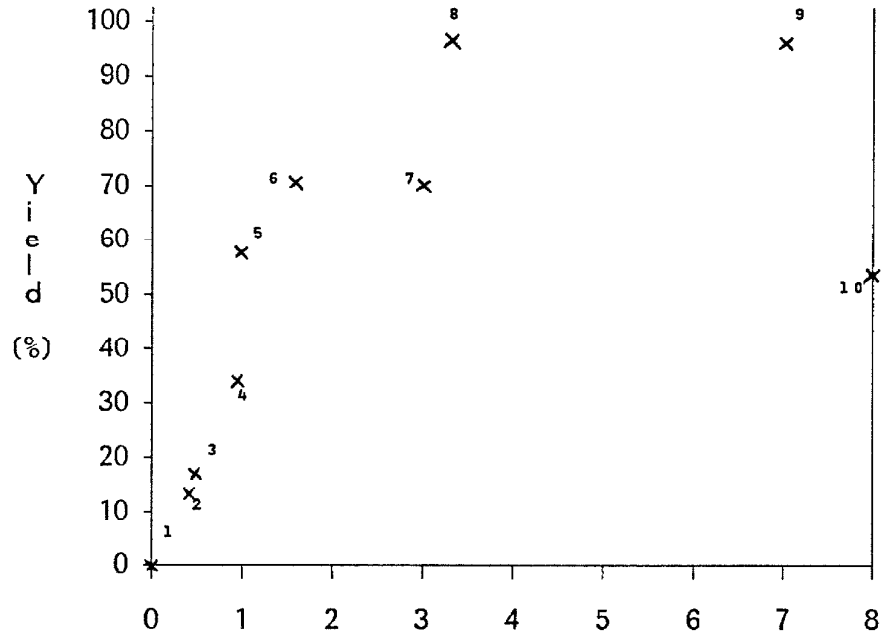


Figure 5.3

Yield as a function of overhead

Figure 5.3 is a scatter plot of yield against overhead for a selection of 64k RAM designs with a defect density of 0.1 per million square lambda. No attempt has been made to fit any of the points to a curve because such a curve would have to represent the maximum possible yield for a given overhead. The points plotted are simply those which can be obtained using Hamming codes and leaf arrays with m times n cells where both m and n are powers of two. The points with overhead less than two indicate that relatively modest amounts of redundancy can be used

to achieve respectable yields for a circuit which would have essentially zero yield with no redundancy. For higher overhead the slope is, of necessity, flatter as the yield approaches 100%.

Well balanced designs

In figure 5.3 there is a band of points which represent designs which are in some way optimal. Other points falling below this band represent designs which fail to provide a high enough yield to justify the overhead. An examination of the bad designs shows that the size, and hence yield, of the leaves is not well suited to the ECC being used. Table 5.1 lists the ECC, number of levels, overhead, leaf array size and calculated leaf yield for each of the designs in figure 5.3. The design using the (5,2) code (number 10) can be seen to be very poor from figure 5.3. Table 5.1 shows that its leaves have a yield of only 14.8%. The surprise then is not that the yield is so low, but that it is as high as 54%.

No.	Code	Levels	Overhead	Leaf size	Leaf yield
1	-	-	0	256x256	0
2	(21,16)	1	0.4	64x64	60%
3	(12,8)	1	0.6	64x128	37.4%
4	(7,4)	1	1	128x128	14.8%
5	(21,16)	2	1	16x16	96%
6	(12,8)	2	1.6	32x32	87.3%
7	(7,4)	2	3	64x64	60%
8	(12,8)	3	3.3	8x16	97.7%
9	(7,4)	3	7	32x32	87.3%
10	(5,2)	2	8	128x128	14.8%

Table 5.1

In chapter 4 there was a discussion of how the YCP could be used to determine suitable leaf sizes for a given code and defect density when using the naive model. Unfortunately there is no easy way of calculating the YCP for the statistical model. Its value varies according to the defect density and the leaf array size in addition to the ECC. An approximate answer for a given set of parameters can be obtained by plotting a set of curves like that in figure 5.1. From figure 5.1 it can be seen that the statistical YCP for the Hamming (7,4) code and a 1k bit leaf is about 70%. The large discrepancy between the calculated value of 94.2% and the experimental value of 70% is another measure of the conservatism of the naive model.

Address space as a function of area

This relationship is hard to demonstrate using results obtained from the statistical experiment because it should be examined at a fixed yield. There are, however, three yields for any given design at which the behavior is known to some extent: 0%, 100% and the YCP. The first two are uninteresting from the point of view of using components with those yields to build larger memories. When the yield of a HRM is equal to the observed YCP for the ECC used, it is possible to make a HRM with k times the address space and the same yield using n + x times the area where x represents the area of the new root node. x will typically be rather less than 1 unless very conservative design rules are used for the root.

This result is probably not very useful in practice but it reflects the theory which states that arbitrarily large memories

can be built using sufficient redundancy. It does provide the ability to predict the yield of large HRMs by looking at the yield of smaller ones. For example, figure 5.3 shows that a 64k RAM using 3 levels of (12,8) redundancy has a yield of 98%. Using 12 of these to build a 512k RAM would result in a memory with nearly 100% yield and approximately 6.5 overhead. In order to make a comparison with the megabit RAM described in chapter 1, it is deduced that a megabit RAM could be built with essentially 100% yield at a overhead of 9. An area overhead of 9 translates into a linear overhead of 3. A megabit irredundant memory using the HRM bit store would be about 10cm square. The HRM would thus be about 30cm square. It is still too large to consider building but could possibly be made small enough by using a process with smaller feature size and decreasing the overhead.

Choosing a design

The choice of parameters for a HRM will depend upon the application for which the memory is required. Figure 5.3 indicates that a HRM with an overhead of between 1 and 3 can enable manufacturers to build memories which have substantially more address space than is possible without redundancy. This type of design is the optimum if density is the most important factor. If reliability is more important, a design with a yield at or close to 100% would be more suitable. Such a design is more likely to produce circuits with enough spare working parts to resist lifetime failures.

The leaf array size, ECC and number of levels can be chosen by looking at the observed YCP for the different ECCs and selecting

a suitable leaf array size. The leaf array size should be chosen to give a yield greater than the observed YCP if high reliability is required and equal to it or somewhat below if low overhead is the goal. Given the ECC and the leaf array size, the number of levels needed to make a memory of the required size is easily determined.

Chapter 6
DESIGN AND ANALYSIS OF THE HRM

Yield and value

For redundant and irredundant circuits the notion of yield as the proportion of circuits which are fully working is a valuable one. For components of redundant circuits the measure is inappropriate because they can make valuable contributions even if they are only partially working. In fact a component with a design error and a yield of zero will be useable provided that its error is removed at a higher level by the redundancy. Thus, when talking about memory components, a more useful measure is the average number of working bits.

In the case of the HRM even this measure is not really good enough because the relationship between the yield of components and the yield of the composite is fairly complex. The result is that components whose average proportion of working bits is higher than the observed YCP are much more valuable than those for which it is not. If one component has a yield less than the YCP then others must have yields higher than the YCP if the overall circuit is to have a yield higher than the YCP. This fact suggests the need for a further measure in which components have a positive or negative value depending upon whether their average is above or below the YCP.

In fact the value of the leaves depends on the failure modes rather than just the average proportion of working bits. This dependency can be demonstrated by an analysis of two special

cases of the single level HRM. In the first case every leaf has exactly one defective bit which can occur at any position with equal probability. In the second case every leaf array has a probability 1/m of total failure where m is the number of bits in the array. In each case the average proportion of working bits is (m-1)/m.

The probability of the memory working in the first case is the probability that no two bits in equivalent positions of different arrays fail or:

$$\begin{cases} \prod_{i=1}^{n-1} (m-i)/m & \text{for } n <= m \\ 0 & \text{otherwise} \end{cases}$$

where the memory uses n leaf nodes. The minimum values of m and n are one and three respectively. For the second case, the probability of working is the probability that no more than one array fails or:

$$((m-1)/m)^{n} + n\,((m-1)/m)^{n-1}\,(1/m)$$

This expression can be rewritten as:

$$((m+n-1)/m)\,((m-1)/m)^{n-1}$$

This result is higher than that obtained for the first case unless m equals one, in which case they are both zero. The first factor is always greater than one and the second factor is clearly greater than the result for the first case if m is greater than one. This result shows that the failure distribution, representing the different failure modes, is important as well as the average number of bits affected.

Single level and multiple level memories

If a HRM has more than one level of redundancy, accurate analysis without the use of statistical modeling is made exceedingly difficult by the mathematical properties of the coding scheme and the effects of partial damage. Since a well-balanced design should have a leaf yield not too much greater than the observed YCP, the use of approximations is highly unreliable. If a good design is evaluated using a pessimistic model, it is likely that the leaf yield will be calculated to be less than the observed YCP and, hence, the overall yield would appear to be very low. The opposite effect would be produced by the use of an optimistic model on a memory whose leaf yield was a little below the observed YCP.

Even in a single level memory the effect of partial damage is hard to calculate. For multiple level memories the problem is compounded by the fact that the errors which propagate up the tree are often the result of the intersections of areas of partial damage. The analysis of multiple level memories other than by statistical means would therefore require very complicated formulae. Appendix C shows the formula for a model used later in this chapter for the analysis of a single level memory. A similar formula for a multiple level memory would be much more complex. Also the formula in Appendix C uses approximations in its treatment of partial damage. These approximations would have to be removed if the formula was to be reasonably accurate.

If a memory has only one level of redundancy then the inaccuracy of optimistic, pessimistic or other approximate models is less

severe. In this case there is some hope of achieving useable results by analytic models and the remainder of this chapter contains a discussion of some possible methods.

Models for single level memories

The following models were investigated in an attempt to determine upper and lower bounds on the yield of a single level HRM. They involve no information about the internal design of the cells but do rely on information about the global wiring structure. For example it is known that data wires run vertically through bit stores and hence that errors within bit stores can affect columns if they affect the data wires. The upper bound is achieved by using optimistic assumptions for the extent of the damage to cells and the lower one by using pessimistic ones.

The lower bound turns out to be given by the naive model; power-to-ground shorts affect the whole array and so all failures are assumed to do likewise. Even if power-to-ground shorts are ignored, no better result is obtained. The occurrence of a defect in a bit cell can cause either a row or column failure and, pessimistically, is assumed always to cause both. Thus, if more than one array is defective, there are at least two addresses within the leaf array for which more than one array has a bad bit. The only improvement over the naive model is that the memory will have fewer bad bits; the yield remains the same. This difference would be useful only in the case of a multi-level memory.

In the least optimistic model errors in bit stores affect only

that bit while errors in row and column decoder cells affect only the appropriate row or column. This model leads to a very complex expression because of the need to calculate the combined effect of multiple defects. The expression can be simplified, and made more optimistic, by ignoring some failures due to mixtures of different types of defect. A further simplification can be achieved by assuming that the effect of failure of individual bit stores is negligible or, equivalently, that bit stores never fail.

When applied to a HRM built from the circuits used for the statistical experiment, these two models result in a very large envelope. For a single level 4k RAM using the (7,4) code and a defect density of 0.6 per million square lambda, the naive model gives a yield of 3.7%, the statistical experiment 47% and the optimistic model 85%. Thus, even for a single level memory, the effect of partial damage cannot be estimated reliably without knowledge of the internal cell designs. Appendix C describes the optimistic model used for this example.

Divided cell approach

The divided cell approach is a mixture of the statistical and analytical models providing greater accuracy than the analytic approximations while requiring less computer time than the full statistical model. Under this approach the cells are initially evaluated using the circles program; they are then divided into sections according to the failure modes and their probabilities. Thus a bit cell in which defects have a 50% probability of affecting one bit, a 20% probability of affecting one row and a

30% probability of affecting one column would be divided into three sub-cells of 50%, 20% and 30% of the area of the original cell.

After all cells have been treated in this way, the new cells are regrouped according to the failure modes which they represent. Thus the proportion of the bit cell which causes a column failure is amalgamated with the proportion of the column address decoder which causes a column failure. The resulting reorganised leaf array is much closer to the idealised form assumed in the analytic models. The analytic models can now be applied using appropriate treatment of the partial column and similar failures to provide optimism or pessimism to give upper or lower bounds. This approach is more accurate than the strictly analytic one because it enables differentiation between the various failure modes which can occur in a single cell.

This approach was applied to the upper bound calculation from the example in the last section. Since the upper bound used an optimistic model, partial row or column failures were treated as single bit failures and multiple row or column failures as single row or column failures. The resulting yield was 78% which is still considerably higher than the 47% given by the statistical experiment. The reasons for this discrepancy are the optimistic treatment of multiple defects in the model (Appendix C) and the treatment of partial and multiple column and row failures.

The experiment was repeated once more where partial row and column failures were treated as a 50% chance of losing a single bit and a 50% chance of losing the whole row or column. Simi-

larly multiple row and column failures were divided equally between single row and column failures and complete array failures. In this case the yield was 63%, showing that the treatment of such cases has a marked effect on the result. Results of this type cannot be used for upper bound calculations because they are no longer uniformly optimistic.

Model name	Bound	Yield
Naive	Lower	3.7%
Statistical experiment	None	47%
Optimistic	Upper	85%
Divided cell optimistic	Upper	78%
Divided cell approximate	None	63%

Table 6.1

Table 6.1 lists the yields given for the example circuit by the various models described. The naive and optimistic models are useful only as a cheap test that a design is within the appropriate yield range. The divided cell models are more expensive because they require the use of the circles program. They do not give a very accurate result if there is a significant proportion of failures involving partial rows or columns or multiple rows or columns but, for designs where such failures are infrequent, these models should give useful results. The statistical experiment is the most expensive model but, for designs like the one analysed here, it is the only model which can provide reasonable accuracy.

Defensive design

The use of redundancy requires a knowledge of the ways in which circuits fail and how to guard against such failures. In irredundant design the circuit is discarded if it does not work properly and it does not matter whether the faults affect large or small parts of the circuit. With redundant design it becomes necessary to study the possible failure modes and their relative frequencies and plan the circuit so that damage is minimised.

An example from the leaf node of the HRM is that a frequent failure mode of the address decoders resulted in one or more pairs of rows or columns being faulty. A simple design change saved one of each pair from being affected thus halving the number of bits affected. Because a large proportion of the failures involving larger numbers of bits were accounted for by this failure mode, the average number of bits lost could be considerably reduced.

Another example was that row and column failures occurred more frequently in those rows and columns furthest from the line drivers. If a uniform addressing scheme was used, that would mean that the addresses corresponding to these positions, and particularly to the corner bit store, would exhibit a rather higher failure rate than the average. By encoding the address decoders in such a way that different leaf arrays have different addresses in those positions, it is possible to even out the distribution of failure probability over the address space.

These two examples illustrate the two main techniques in defensive design. The first is that of design changes which reduce

the amount of circuitry affected by a failure. The second does not reduce the damage but serves to reduce its effect on the performance of the overall circuit.

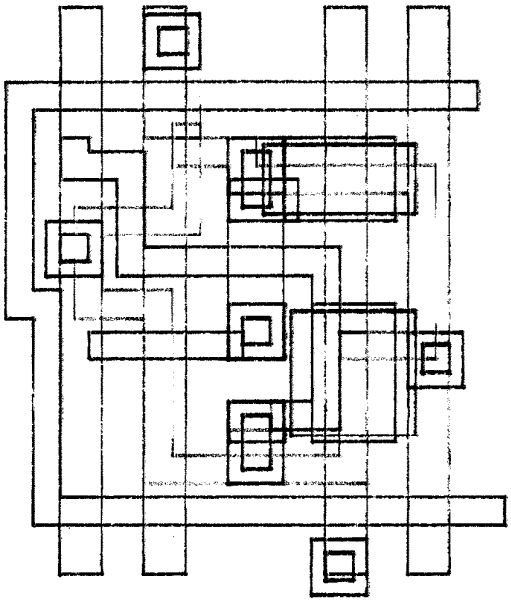


Figure 6.1

Defensive design can be used to make the analytic models more realistic. In the case of the HRM it would be possible to design the cells using duplicate wires in certain locations. Figure 6.1 shows a bit store cell which uses a double select line to reduce the probability of partial row failure. Partial row failures cannot occur without either a very large single defect or two suitably located normal sized defects. By using similar techniques in the row and column decoders for protection against

address line failures, the circuit could be made approximate to one in which a defect in a bit store caused single bit, partial column or full column failure and a defect in a row or column decoder caused single row or column failure. An envelope could then be obtained by assuming that all bit store defects caused column failures for a pessimistic model and single bit failures for an optimistic model. It is important to note that while such strategies improve yield and aid analysis they are not necessarily the best strategies in terms of yield improvement versus area cost.

Iterative design

The circles program can be used as a basis for iterative design. The major design flaws in a circuit can be identified in the early iterations; later ones would allow the designer to experiment with design changes which affect the proportions of different failure modes.

	Old design	New design
% of circles causing defects	26.3	24.5
Area in square lambda	1116	1216
Relative area	1	1.09
Single bit failures	43%	51.5%
Double bit failures	7%	1%
Part row failures	11%	-
Part column failures	28%	40%
Double part column failures	-	2%
Whole column failures	3%	5.5%
Whole array failures	8%	-

Table 6.2

The bit cell design used in the statistical experiment was designed to be as small as possible. A major design flaw was that regions of diffusion carrying power and ground were close enough to each other that defects could cause shorts. Figure 6.1 shows a revised design in which power and ground were kept further apart to prevent shorts and the select line was replicated to reduce the probability of partial row failures. The result was a larger cell with more open space which was affected by a smaller proportion of possible defects and sustained less damage per defect. Table 6.2 gives a comparison of the two cells.

Where leaf arrays contain s columns of s bits each, the average number of bits lost per 100 defects in the cell array is

$$8 s^2 + 22.5 s + 57$$

for the original design and

$$27.5 s + 53.5$$

for the new design. These formulae are derived by multiplying the number of bits lost due to failure by the probability of that type of failure; partial row and column failures are counted as affecting s/2 bits. Table 6.3 shows the evaluation of these expressions for three different values of s. The absence of a term for s squared in the expression for the new design means that, for typical array sizes, the new design results in several times fewer bad bits. The increase in cell area is about 9%.

For irredundant designs the normal practice is to minimise the area occupied by a cell while not violating the design rules. In fault tolerant designs extra area can be used either to implement

s	Old design	New design
1	87.5	81
8	749	273.5
32	8969	933.5

Table 6.3

extra functions or to reduce the probability of error in existing functions. After it has been decided how much area to allot to a cell, it is still important to experiment with layout topology. In the above example of the redesigned bit store, the reduction in the average number of bad bits was achieved by topological changes. Iterative design using the circles program is an effective way of evaluating and improving cell designs. Designers of irredundant circuits accept a cell design when the effort of making it smaller becomes too high; designers of redundant circuits would accept a cell design when the effort of reducing its susceptibility to errors became too high.

Chapter 7
CONCLUSIONS

Timeliness

The use of redundancy is not in itself a new idea nor is the possibility of using it in integrated circuit design. It is therefore necessary to examine why it has not been widely used when it would appear from the above results that it is eminently feasible and, for many applications, economically advantageous. The most obvious answer is that there has been very little work done on investigating the benefits and limitations of redundant design. As can be seen from the preceding chapters, there is no simple relationship between redundancy and yield.

Another factor is that the size of circuits is becoming limited by the size of the well in the standard integrated circuit package. Any attempt to include redundancy would increase the size of the circuit, preventing it from fitting in the standard package, or decrease the function, lowering the price that can be commanded for the part. Soon it will be necessary to start using larger packages for irredundant circuits anyway and it is to be hoped that a standard allowing the use of redundant circuits will be adopted. It is interesting to note that IBM, the producer of a 64k RAM using static redundancy, does not use the industry standard package even for its irredundant circuits.

In the case of static redundancy, the tradeoffs are fairly obvious and the departure from traditional design is relatively small. The hardest problems are the design of the configuration

circuitry and the choice of the optimum number of spare modules.

For dynamic redundancy there is more reason for its absence from the marketplace as, until recently, it was not economical. In order to build a HRM with a high enough yield, it is necessary that the yield of the bottom level decoder be fairly close to 100%. It is now possible to build a Hamming (7,4) encoder/decoder and a 4-way multiplexor with a combined yield of about 96-98%. This yield is barely sufficient to enable the building of a HRM because, for the (7,4) code, the YCP is just over 94%. Since the failure of a non-leaf node affects the whole sub-tree of which it is a root, the non-leaf node must have a yield greater than the YCP for its ECC by a sufficient margin for some errors in its children to be corrected. The HRM has thus only recently become a cost-effective alternative to traditional designs. As processing techniques improve the cost-effectiveness of the HRM will increase greatly because the yield of the non-leaf nodes will get higher, enabling larger leaf arrays with correspondingly more errors to be used.

Defensive design

In order to reduce damage effectively it is necessary to have a good understanding of the possible failure modes. This understanding can only be achieved by the building of test circuits and the application of suitable electrical tests and visual inspections. Different processing technologies are susceptible to different failure modes; it is possible that the dominant technology in the future will be the one which is best suited to minimisation of damage.

The results of the statistical experiment show that, even without much regard to defensive design, the HRM can be used to increase yield of existing memories and enable the building of larger ones. The use of suitable defensive design techniques will result in smaller overheads than is indicated by this research.

Possible refinements to the HRM

The description of the HRM given above is the one used in this research. There are some refinements which could be made to improve various aspects of the memory according to the designer's requirements. These refinements are concerned with the addition of static redundancy or extra processing capability to improve yield or decrease testing time or both. The reason for not using these techniques in the analysis presented here is that the intention is not to produce an optimum design but to demonstrate the feasibility of a class of designs.

The addition of static redundancy by means of including extra leaf nodes or extra storage within the leaves or both can be used to reduce susceptibility to fabrication flaws. In the basic design the loss of a complete leaf array causes any other failures in the group of leaves to result in an incorrect bit being passed up to the next level. In the case of two array failures in the same group all bits in the group are rendered useless. The inclusion of an extra leaf in each group would mean that if only one array failed it could be discarded with a resulting high probability that the sub-tree would work correctly for all addresses. In the case of a double array failure, one of the failed arrays would be discarded with the result that, in

most cases, a reasonable proportion of the sub-tree would work and so make a useful contribution to the next level. If no leaf arrays fail completely whichever one has the most errors or, better, whichever one has the most errors coinciding with errors in other leaves in the group, would be discarded. If three arrays failed, it would be impossible to pass up corrected bits to the next level, but each bit position in the multiplexor could be connected to one of the partially working arrays thus ensuring that at least some of the bits passed up were working correctly. This stratagem could also be applied to the basic HRM without static redundancy in the case where two or more arrays fail.

The problem with the inclusion of static redundancy is that extra logic has to be included in the circuit to enable testing of physical addresses in the memory and configuration according to the results of the testing. In addition to the increased testing time there is the problem of failure of the testing and configuration circuitry. For example a leaf array might work perfectly but its physical address mechanism be faulty. In that case the testing will show the array to be useless when in fact it is not. Conversely it could happen that testing shows the array to be fully working but the configuration circuitry is faulty, resulting in an attempt to use a good array through a faulty switch. Despite these problems it would appear that the benefits possible from such techniques could be considerable and that this area is worthy of further study.

The addition of extra storage within an array would at first sight not appear to be very valuable. Such extra storage can be used for protection only against small numbers of errors in each

array and the whole memory works on the basis of guarding against small numbers of errors. However, if one array in a group fails completely then all other errors in arrays in the same group will result in errors being passed up to the next level. If such errors occur in small numbers they can be corrected by using spare bits or lines of bits. Thus small numbers of extra bits can be used to help correct the errors due to array failures. Again this type of redundancy involves the testing complications described for the inclusion of extra arrays.

The addition of extra processing power in the non-leaf nodes is a very attractive idea which could reduce testing time dramatically and might be used to overcome the problems outlined above for static redundancy. It is not currently practicable because of the need to maintain a high yield for non-leaf nodes and it is now impossible to build significantly more complex non-leaf nodes than the simple form which performs the minimum function described above. In the future it is likely to be possible to include a moderately powerful, but probably serial, processor in each such node. One of the main tasks of this processor would be to test its sub-tree, including any configuration circuitry, and configure it to provide the maximum possible number of corrected bits. This modification would substantially decrease testing time because external test equipment would be necessary only for testing the root node. If the root was found to be faulty the circuit would be discarded, otherwise it would be set to test itself. This testing would not only require a minimum of special equipment, but would proceed more quickly than external testing because it could be done in parallel in the different sub-trees

resulting in a logarithmic time complexity.

Conclusions

The results of the statistical experiment show that redundant design of memories can be cost-effective without the use of special processes for providing fuses and switches. Moreover such designs provide valuable protection against transient errors and circuit failures. The circuit used for the statistical experiment was designed without much regard to defensive design. The experiment described in the section about iterative design indicates that, with suitable tools, it is possible to make major improvements in cell designs at very little cost in area. It should therefore be possible to build HRMs with substantially higher yields than indicated by the statistical experiment.

Appendix A
TABLES OF RESULTS

Meaning of the tables

The following tables list the results obtained for a number of different designs at three memory sizes and six defect densities. Each result was obtained from "building" 100 memories. These numbers were used as an indication of which examples to use in chapter 6. A further sample of 1000 was taken to provide the data used in chapter 6.

0.1 faults per million square lambda

Code	Levels	Overhead	Yields: 4k, 16k, 64k		
-	-	0	56%	14%	0%
(21,16)	1	0.4	91%	58%	14%
(12,8)	1	0.6	91%	65%	17%
(7,4)	1	1	86%	73%	34%
(21,16)	2	1	-	77%	58%
(12,8)	2	1.6	99%	97%	72%
(7,4)	2	3	100%	94%	71%
(12,8)	3	3.3	-	99%	98%
(7,4)	3	7	100%	100%	98%
(5,2)	2	8	-	-	54%
(7,4)	4	15	-	100%	100%

0.2 faults per million square lambda

Code	Levels	Overhead	Yields: 4k, 16k, 64k
-	-	0	35% 1% 0%
(21,16)	1	0.4	73% 33% 1%
(12,8)	1	0.6	80% 37% 6%
(7,4)	1	1	88% 48% 24%
(21,16)	2	1	- 52% 6%
(12,8)	2	1.6	92% 81% 21%
(7,4)	2	3	98% 82% 26%
(12,8)	3	3.3	- 90% 73%
(7,4)	3	7	100% 100% 89%
(5,2)	2	8	- - 32%
(7,4)	4	15	- 100% 100%

0.3 faults per million square lambda

Code	Levels	Overhead	Yields: 4k, 16k, 64k
-	-	0	30% 0% 0%
(21,16)	1	0.4	53% 16% 0%
(12,8)	1	0.6	59% 25% 4%
(7,4)	1	1	68% 40% 13%
(21,16)	2	1	- 18% 0%
(12,8)	2	1.6	77% 47% 1%
(7,4)	2	3	95% 64% 11%
(12,8)	3	3.3	- 58% 17%
(7,4)	3	7	100% 95% 59%
(5,2)	2	8	- - 17%
(7,4)	4	15	- 100% 100%

0.4 faults per million square lambda

Code	Levels	Overhead	Yields: 4k, 16k, 64k		
-	-	0	11%	0%	0%
(21,16)	1	0.4	45%	8%	0%
(12,8)	1	0.6	50%	15%	3%
(7,4)	1	1	54%	26%	13%
(21,16)	2	1	-	6%	0%
(12,8)	2	1.6	70%	27%	0%
(7,4)	2	3	88%	42%	2%
(12,8)	3	3.3	-	25%	3%
(7,4)	3	7	100%	90%	22%
(5,2)	2	8	-	-	12%
(7,4)	4	15	-	100%	98%

0.5 faults per million square lambda

Code	Levels	Overhead	Yields: 4k, 16k, 64k		
-	-	0	9%	0%	0%
(21,16)	1	0.4	35%	4%	0%
(12,8)	1	0.6	35%	10%	1%
(7,4)	1	1	53%	20%	8%
(21,16)	2	1	-	4%	0%
(12,8)	2	1.6	53%	18%	0%
(7,4)	2	3	88%	26%	0%
(12,8)	3	3.3	-	8%	0%
(7,4)	3	7	98%	75%	6%
(5,2)	2	8	-	-	4%
(7,4)	4	15	-	100%	80%

0.6 faults per million square lambda

Code	Levels	Overhead	Yields: 4k, 16k, 64k		
-	-	0	3%	0%	0%
(21,16)	1	0.4	29%	3%	0%
(12,8)	1	0.6	38%	5%	1%
(7,4)	1	1	47%	16%	7%
(21,16)	2	1	-	1%	0%
(12,8)	2	1.6	43%	6%	0%
(7,4)	2	3	71%	13%	0%
(12,8)	3	3.3	-	1%	0%
(7,4)	3	7	90%	59%	1%
(5,2)	2	8	-	-	7%
(7,4)	4	15	-	94%	49%

Appendix B

DERIVATION OF EQUATION 4.1

Equation 4.1 was claimed to be an expression for the probability of a (7,4) one level HRM working when the simple model treats it as not working. The equation given was:

$$P = \sum_{i=2}^{64} a\, b^{7}\, b^{448-i}\, (1-b)^{i}\, C_{i}^{64}\, (7^{i} - 7)$$

The summation is for varying numbers of bit failures. The minimum is 2 because the case of one failure is covered by the simple model. The maximum is 64 because the memory cannot work if more than 64 bits fail. The term in a represents the case that all 7 decoders work. The terms in b and 1-b give the probability of a given pattern of i bits failing. The remaining terms are a count of such failure modes. The first one is the number of ways of allocating the bits to specific addresses within the arrays. The second is the number of ways of allocating the bits among the seven arrays such that not all are in the same array.

Appendix C

AN OPTIMISTIC MODEL FOR THE SINGLE LEVEL HRM

Definitions

y = yield of overall circuit
ye = yield of all except non-leaf node
yb = yield of bit store cells
yr = yield of row decoder cells
yc = yield of column decoder cells
yd = yield of non-leaf node
n = number of leaf nodes
s = height and width of leaf array (in bits)
m = number of bits in leaf array (= s squared)

Failure groups

The yield of the whole memory (y) equals the product of the yields of the non-leaf node (yd) and the remainder of the circuit (ye). There are eight possible groups of failures: 1) no failures, 2) column failures, 3) row failures, 4) bit failures, 5) row and column failures, 6) column and bit failures, 7) row and bit failures, 8) row and column and bit failures. For each case the probability of that case occurring without causing overall failure is calculated.

Case 1

The probability of no failures is:

$$yc^{ns} \quad yr^{ns} \quad yb^{nm}$$

probability that all the row decoders and bit stores work and that no more than s columns fail with no two failed columns being in the same position of different arrays:

$$yr^{ns}\, yb^{nm} \sum_{i=1}^{s} n\, C_i\, yc^{ns-i}\, (1-yc)^{i}$$

Case 3

The probability of benign row failures is similar to case 2 with yr and yc interchanged:

$$yc^{ns}\, yb^{nm} \sum_{i=1}^{s} n\, C_i\, yr^{ns-i}\, (1-yr)^{i}$$

Case 4

The probability of benign bit store failures is the probability that not more than m bits fail, with no failures occurring in the same address in different arrays:

$$yc^{ns}\, yr^{ns} \sum_{i=1}^{m} n\, C_i\, yb^{nm-i}\, (1-yc)^{i}$$

Case 5

For a mixture of row and column failures to be benign, they must all occur in the same array. The probability is derived by taking the case that, for one array, not all rows and not all columns work:

$$yb^{nm}\, yc^{(n-1)s}\, yr^{(n-1)s}\, n\, (1-yc^{s})\, (1-yr^{s})$$

Case 6

This case is simplified by taking the probability that none of the column failures conflict with each other and that none of the bit failures conflict with each other. The case of column failures conflicting with single bit failures is ignored:

$$yr^{ns}\left(\sum_{i=1}^{s} {}^{n}C_{i}\, yc^{i}(1-yc)^{ns-i}\right)\left(\sum_{i=1}^{m} {}^{n}C_{i}\, yb^{i}(1-yb)^{nm-i}\right)$$

Case 7

This case is derived from case 6 by interchanging yc and yr:

$$yc^{ns}\left(\sum_{i=1}^{s} {}^{n}C_{i}\, yr^{i}(1-yr)^{ns-i}\right)\left(\sum_{i=1}^{m} {}^{n}C_{i}\, yb^{i}(1-yb)^{nm-i}\right)$$

Case 8

This is similar to case 5 where all failures have to be in the same array:

$$yb^{(n-1)m}\, yr^{(n-1)s}\, yc^{(n-1)s}\, n\,(1-yb^{m})\,(1-yc^{s})\,(1-yr^{s})$$

Calculation of yield

The yield for the whole memory is given by

$$y = yd\ ye$$

where ye is obtained by taking the sum of the probabilities of the eight cases.